Questions 1-10 are multiple choice (4 pts each). Select the answer that best represents the output of the code fragment. Circle the letter of the answer on the answer sheet.

**Question 1**

```python
def nthInstance(string, char, n):
    string = string.lower()
    position = 0
    instance = 0
    for chr in string:
        if chr == char:
            instance += 1
            if instance == n:
                return position
        position += 1
    return -1

s = 'A man a plan panama'
print(nthInstance(s, 'a', 2))
```

a. -1
b. 0
c. 3
d. 6
e. none of the above

**Question 2**

```python
def trailers(s):
    rtnVal = ''
    sList = s.split()
    for word in sList:
        if word[-1] not in rtnVal:
            rtnVal += word[-1]
    return rtnVal

t = "If the facts don't fit the theory, change the facts."
print(trailers(t))
```

a. fest
b. fest,
c. festte,ee.
d. SyntaxError: illegal expression for augmented assignment
e. none of the above
Question 3
def matchDoubleMatch(t, letter, length):
    tList = t.split()
    match = t.count(letter)
    doubleMatch = 0
    for word in tList:
        if len(word) == length and letter in word:
            doubleMatch += 1
    return [match, doubleMatch]
s = 'behold the mighty dinosaur, famous in prehistoric lore'
print(matchDoubleMatch(s, 'h', 6))
a. [0, 2]
b. [4, 2]
c. [4, 3]
d. Syntax error: two return values given, one expected
e. none of the above

Question 4
hca = ['Once', 'upon', 'a', 'time', 'there', 'were', 'three', 'billy', 'goats']
index = 0
while index < len(hca):
    if len(hca[index]) != 4:
        index += 1
    continue
    index += 10
print(index)
a. 0
b. 2
c. 9
d. 10
e. none of the above

Question 5
chars = [{1:'Dumbledore'},{'profs':['Snape','McGonagall']},['Ron', 'Hermione'], 'Harry']
print((chars[1][0]))
a. Dumbledore
b. Snape
c. 'p'
d. KeyError: 0
e. none of the above
Question 6

profound = "It's not who I am underneath, but what I do that defines me"

def extremeSymmetryCount(t):
    count = 0
    lst = t.split()
    for word in lst:
        if word[-1] == word[0]:
            count += 1
    return count

print(extremeSymmetryCount(profound))

a. 1
b. 2
c. 3
d. IndexError: string index out of range
e. none of the above

Question 7

jingle = [1:'money', 2:'show', 3:'get ready', 4:'go']

print(jingle[0])

a. KeyError: 0
b. 1
c. 1:'money'
d. 1:'money', 2:'show', 3:'get ready', 4:'go'
e. none of the above

Question 8

bools = [True, not False, not not True, not False and True, not True or False]

trueCount = 0

for expr in bools:
    if expr == True:
        trueCount += 1

print(trueCount)

a. 1
b. 2
c. 3
d. 4
e. none of the above
**Question 9**

```python
import turtle
s = turtle.Screen()
t = turtle.Turtle()
for i in range(4):
    if i%2 == 0:
        t.fd(100)
    else:
        t.right(90)
```

a. a straight line  
b. two perpendicular lines  
c. three sides of a square  
d. a square  
e. none of the above

**Question 10**

```python
wishesWereFishes = True
wishesWereHorses = False
if wishesWereFishes:
    print('beggars eat')
if wishesWereHorses:
    print('beggars ride')
else:
    print('beggars walk')
```

a. beggars eat  
b. beggars walk  
c. beggars eat  
  beggars walk  
d. SyntaxError: invalid syntax  
e. none of the above
Questions 11, 12 and 13 are programming problems. Put your solutions on the ruled answer sheets. Use the vertical rules to show the level of indentation of the code.

**Question 11A**
Write a function named rectangle() that uses turtle graphics to draw a rectangle. The function rectangle() takes two parameters:

i. \( t \), a turtle that is used to draw and
ii. \( shortSide \), the length of a short side of the rectangle

The ratio of the length of a short side to a long side is 1:2.

The function rectangle() should draw a short side first. (Hint 1: the turtle must be down to draw.) It should leave the turtle at the same location and with the same orientation it had initially. (Hint 2: the turtle may initially be at any location on the screen and in any orientation.) For full credit, you must use a loop to perform the repeated operations.

**Question 11B**
Write a function named adjacentRectangles() that draws a series of rectangles next to each other. Each rectangle is double the size of the preceding one. The function adjacentRectangles() takes three parameters:

i. \( t \), a turtle that is used to draw and
ii. \( shortSide \), the length of the short side of a rectangle, and
iii. \( number \), the number of rectangles to draw

The function adjacentRectangles() should call the function rectangle() that you coded in 11A to draw each rectangle.

For example, suppose that adjacentRectangles() is called by the following code:

```python
import turtle
s = turtle.Screen()
t = turtle.Turtle()
size = 10
numSides = 5
adjacentRectangles(t, numSides, size)
```

Then this would be correct graphical output.
**Question 12**

Write a function named `repeatCount()`. The function `repeatCount()` takes two string parameters: the name of an input file and the name of an output file. The function `repeatCount()` should read each line of the input file, identify the number of words on the line that occur more than once, and write that number to a line in the output file. A word should be considered to be repeated if occurs with different capitalization. That is, 'To' and 'to' are the same word. You may assume that the file contains only upper and lower case letters and white space; it does not contain any punctuation marks.

For example, suppose the input file contains the following four lines:

*Woke up this morning with an ache in my head*
*I splashed on my clothes as I spilled out of bed*
*I opened the window to listen to the news*
*But all I heard was the Establishment Blues*

Then the output file should contain the following four lines:

0
1
2
0

**Question 13**

Write a function named `wordPositions()` with the following input and output.

**Input:** $s$, a string consisting of upper and lower case letters and spaces.

**Return:** a dictionary in which each distinct word in $s$ is a key and the corresponding value is a list of the positions in $s$ in which the word occurs. Words are to be treated as the same regardless of their capitalization. That is, "Yes" and "yes" are the same word.

The following is an example of correct output.

```python
>>> s = 'One fish two fish red fish blue fish'
>>> wp = wordPositions(s)
>>> print(wp)
{'two': [2], 'one': [0], 'red': [4], 'fish': [1, 3, 5, 7], 'blue': [6]}
```